REMARKS/ARGUMENTS

Reconsideration and withdrawal of the rejections set forth in the Office Action dated April 5, 2006, are respectfully requested. The claims have been amended for the sole purpose of clarity. Claim 13 is new. Claims 1-13 are currently pending this application.

THE 102(e) REJECTIONS

The Examiner rejected claims 1-12 under 35 U.S.C. 102(e) as anticipated by U.S. Pat. No. 6,374,402 (Schmeidler et al.).

THE PRIOR ART

There are two important points that the Examiner must understand regarding file system hooks and overlays:

1) A file system hook is used to intercept calls to a local existing file system (e.g., on the C drive). However, if you implement a file system driver, there is no need to hook because the operating system is already directing file system calls to your driver. Notably, Schmeidler et al. never refer to file system hooks because they have no need for them.

2) A file system overlay lets you mix local files and streamed files. An overlay makes it look like the streamed application is actually on a local existing file system. With Schmeidler et al., the application is run on the virtual drive, in its own separate file system.

Schmeidler et al. use virtual drives. They say they use a V-drive throughout the reference. A briq file contains a representation of a file system (see, e.g., FIG. 12). The ARFSD, which is a file system driver, creates the illusion that the contents of the briq file is a locally mounted file system (see, e.g., col. 16, lines 9-33). The streamed application is then run from that file system. If the path associated with a file includes the pseudo-network driver, Schmeidler et al. do not have to hook anything because the operating system will direct it automatically to their driver; there is nothing to hook. In addition, Schmeidler et al. do not use a file system overlay because local and streamed files are not mixed together.

The Examiner indicates at page 2 of the Office Action that Schmeidler et al. teach at col. 3, lines 1-6, a file system hook. Notably, Schmeidler et al. do not refer to a file system hook at this section, or anywhere else in the reference. Indeed, at col. 2, line 47 to col. 3, line 6, Schmeidler et al. explain:

> Upon completion of the purchase negotiation, SCDP client software running on the user's PC obtains an authorization token and keying material from a Conditional Access Server (CAS). The token authorizes the client process to run the selected title from a network file server accessible across the broadband network. The data retrieved from the file server is encrypted. The SCDP client process uses the keying material provided by the conditional access server to decrypt the data from the file server. With the present invention, titles run on the user's PC, but the title is not downloaded, in its entirety, onto the PC. A title is formatted into an electronic package that contains the title's files in a compressed and encrypted form, referred to hereafter as a briq. ***The briq is actually a portable, self-contained file system, containing all of the files necessary to run a particular title.*** Briqs are stored on a network file server, referred to hereafter as a RAFT server, accessible across a broadband network. The SCDP client treats the briq like a local file system on the user's PC. ***When running a title, the operating system, e.g. Windows, makes read requests to this local file system.*** The SCDP client, which, in the illustrative embodiment, includes a Windows Virtual Device Driver (VxD), services these requests by retrieving the requested blocks of briq data from the RAFT server. After retrieving the requested block of data, the VxD decompresses and decrypts the briq data, and passes the data onto the operating system on the user's PC. (Emphasis added).

Thus, Schmeidler et al. describe a self-contained file system, which obviates use of a file system hook. They have their own file system driver so there is no file system hook, and there is no file system overlay. In fact, the operating system makes read requests directly to the local file system associated with the briq.

THE PRIOR ART DISTINGUISHED

Claim 1 includes the language, "a file system hook operatively interposed between a file system and an operating system of a computer, the file system hook configured to detect a file system call associated with a streamed target program and to perform one or more procedures." To anticipate a claim, a reference must teach each and every element of the claim. Since Schmeidler et al. do not teach a file system hook, claim 1 is allowable over the reference.

Claims 2-8 and 13, which depend from claim 1, are allowable at least for depending from an allowable base claim, and potentially for additional reasons. For example, claim 2 includes a file system overlay, which Schmeidler et al. do not disclose.

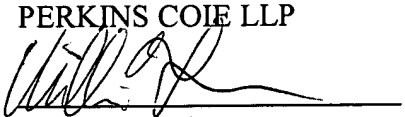
Claims 9-12 are allowable for similar reasons to those described above with reference to claims 1-8 and 13.

CONCLUSION

In view of the foregoing, Applicant submits that the claims pending in the application patentably define over the prior art. The Applicant respectfully requests the Examiner withdraw rejections of all claims. A Notice of Allowance is respectfully requested.

If in the opinion of the Examiner, a telephone conference would expedite the prosecution of the subject application, the Examiner is encouraged to call the undersigned at (650) 838-4305.

Respectfully Submitted,
PERKINS COIE LLP



Date: September 22, 2006

William F. Ahmann
Reg. No. 52,548

Correspondence Address:
Customer No. 22918
Perkins Coie LLP
P.O. Box 2168
Menlo Park, CA 94026
650-838-4300